

# Lowering the error floor of Gallager codes: a statistical-mechanical view

**Marco Pretti**

Consiglio Nazionale delle Ricerche – Istituto dei Sistemi Complessi (CNR–ISC)  
@ Dipartimento di Scienza Applicata e Tecnologia (DISAT), Politecnico di Torino,  
Corso Duca degli Abruzzi 24, I-10129 Torino, Italy  
E-mail address: [marco.pretti@polito.it](mailto:marco.pretti@polito.it)

**Abstract.**

The problem of error correction for Gallager’s low-density parity-check codes is famously equivalent to that of computing marginal Boltzmann probabilities for an Ising-like model with multispin interactions in a non-uniform magnetic field. Since the graph of interactions is locally a tree, the solution is very well approximated by a generalized mean-field (Bethe–Peierls) approximation. Belief propagation (BP) and similar iterative algorithms are an efficient way to perform the calculation, but they sometimes fail to converge, or converge to non-codewords, giving rise to a non-negligible residual error probability (error floor). On the other hand, provably-convergent algorithms are far too complex to be implemented in a real decoder. In this work we consider the application of the *probability-damping* technique, which can be regarded either as a variant of BP, from which it retains the property of low complexity, or as an approximation of a provably-convergent algorithm, from which it is expected to inherit better convergence properties. We investigate the algorithm behaviour on a real instance of Gallager code, and compare the results with state-of-the-art algorithms.

## 1. Introduction

In recent years, new overlaps have emerged between statistical mechanics and other fields of scientific and technical research. In this framework, an important role has been played by *belief propagation* (BP), that is, a class of *message-passing* algorithms, suited to solving several types of constraint-satisfaction and statistical-inference problems [1, 2]. It is straightforward to show that such problems, usually defined via graphical models, can generally be mapped onto proper thermodynamic models of the Ising or Potts type, and ultimately amount to computing marginals of the Boltzmann distribution of the latter [1, 2, 3, 4]. A very important problem of this kind is that of error correction for digital transmission over a noisy channel (*channel decoding*). Here we shall focus in particular on the so-called *low-density parity-check* (LDPC) codes, which exhibit extremely good (i.e. capacity-approaching) performance in the limit of infinite block length, and which have therefore attracted great attention from communication engineers [5], albeit quite long after Gallager’s original proposal [6]. Indeed, BP as an efficient, though suboptimal, iterative decoding algorithm was first proposed (and simply called *probabilistic decoding*) by Gallager himself, who also realized that its underlying approximation breaks down when the graph describing the code structure includes cycles. In the statistical-mechanical analogue, the inference problem arising from Gallager codes can be viewed as an Ising model with multispin interactions in a non-uniform magnetic field [7] (the infinite block-length limit corresponding to the thermodynamic limit), while BP equations turn out to be equivalent to the self-consistency equations derived by the Bethe–Peierls approximation [8, 9]. The latter, well-known in statistical mechanics [10], is a refined mean-field theory, aimed at improving the simpler (Bragg–Williams) mean-field approximation. Note that the Bethe–Peierls approximation dates back to 1935, but the equivalence with BP is a relatively recent result: it was suggested in 1998 by Kabashima and Saad [11], and was proved in a rather general setting a few years later by Yedidia, Freeman and Weiss [12]. Even though the Bethe–Peierls approximation is rigorously exact only on tree graphs (i.e. graphs without cycles), it was found to be extremely effective even on “loopy” graphs [13], in particular when the latter are sufficiently sparse and random (and therefore locally tree-like), as is the case for Gallager codes.

A major drawback of the BP algorithm is its possible lack of convergence. Such a behaviour is sometimes a signal of intrinsic inadequacies of the Bethe–Peierls approximation, which therefore deserves further improvements [4, 12]. If this is not the case, it may be useful to devise strategies to overcome convergence difficulties, and, in order to do so, one can exploit the fact that the Bethe–Peierls approximation admits a variational formulation (usually regarded as an instance of Kikuchi’s cluster-variation method) [4, 14]. The variational formulation states [1, 2, 3, 4, 12] that BP fixed points are stationary points of a variational functional, called *Bethe free energy* (more precisely, it is known that *stable* fixed points of BP are local *minima* of the Bethe free energy [15]). As a consequence, the problem can be addressed by methods that are

able to minimize the Bethe free energy. One such method has been proposed by Heskes, Albers and Kappen in the form of a double-loop algorithm, performing a sequence of partial optimizations, which can be proved to converge to a local minimum [16]. Unfortunately, a double-loop strategy is often practically unfeasible, because of the very long computation time required. For this reason, in a previous paper we proposed a single-loop iterative algorithm, which can be regarded either as an approximation of the aforementioned double-loop algorithm, or as a modified BP algorithm with over-relaxed dynamics, and from there it is denoted as *probability-damping belief propagation* (PDBP) [17]. The latter algorithm (actually, a set of algorithms, specified by an adjustable *damping parameter*) can be verified empirically to be more stable and faster than ordinary BP [17, 18, 19], with just a slightly larger numerical complexity. Note that the interpretation of PDBP as an approximation of the double-loop algorithm is relevant to get a qualitative understanding of how it works and also, more practically, to drive the choice of the damping parameter value.

In this paper, we shall investigate the behaviour of PDBP (and a further variant) as a decoding algorithm on a real instance of Gallager code. The motivation behind this analysis is that the performance of real (finite length) Gallager codes under BP decoding (and other iterative algorithms as well) turns out to be significantly worse than expected, with the onset of a non-negligible residual error probability, even in the practically relevant regime of high signal-to-noise ratio [20, 21, 22]. Such a phenomenon, called *error-floor* in the jargon of coding theorists, is mostly ascribed to specific configurations of the channel errors (called *trapping sets* or the like), which hamper the convergence of the decoding algorithm. Due to the technical importance of the subject, great efforts in research have been made to solve this problem, following mainly two different routes, namely, either eliminating trapping sets by changing the graphical structure of the code (usually removing short cycles) [23, 24], or modifying the algorithm dynamics, in order to make it less sensitive to trapping sets [25, 26]. Along the latter line, considered in this paper, several algorithms have been proposed, among which a noticeable case is the *difference-map belief propagation* (DMBP) algorithm by Yedidia, Wang and Draper [26]. Since this algorithm exhibits significant performance improvement with respect to standard BP approaches, with a moderate increase in complexity, we shall use it as a benchmark for evaluating our algorithms.

The paper is organized as follows. In section 2 we briefly review the formal analogy between a Gallager code and a multispin Ising model. As a model of a noisy channel, we consider the simple *binary symmetric channel* (BSC), even though the analogy holds even for more realistic models such as the *additive white Gaussian noise* (AWGN) channel. In section 3 we introduce the BP algorithm and its variants for the model of interest, using the specific formulation suitable for binary variables and usually denoted as *effective fields* or *log-likelihood ratios* (in statistical mechanics or coding theory, respectively). Section 4 is the central section, where all the results of our simulations are illustrated and discussed. Section 5 contains some concluding remarks.

## 2. Gallager codes and multispin Ising model

Let  $\mathbf{s} \equiv [s_1, \dots, s_N]$  denote an encoded sequence of binary digits (bits), transmitted over a BSC. According to a parity-check coding scheme, such a sequence cannot be any possible sequence of  $N$  bits, but it must indeed satisfy some prescribed parity checks. A parity check is defined by a (usually small for LDPC) subset of indices  $a \subseteq \{1, \dots, N\}$ , so that the corresponding set of transmitted bits  $\{s_i\}_{i \in a}$  must include an even number of 1 bits. A particular code is specified by the set  $\mathcal{P}$  of parity checks, each one represented by a given set of indices  $a$ , while the number  $|\mathcal{P}|$  of (independent) parity checks determines the *rate* of the code  $\rho = 1 - |\mathcal{P}|/N$ . A sequence  $\mathbf{s}$  that satisfies all the required parity checks  $a \in \mathcal{P}$  is called a *codeword*. The structure of a code can also be represented by a bipartite graph, whose two node classes are associated to bits and parity checks (labelled respectively by  $i$  and  $a$ ), with a link whenever  $i \in a$  (i.e. when the bit  $i$  is involved in the parity check  $a$ ). Such a graph is usually called a *Tanner graph*, or a *factor graph* in more general contexts [27]. As previously mentioned, for LDPC codes the factor graph is sparse and locally similar to a tree.

The memoryless BSC flips each bit independently with a given probability  $x$ , so that the received sequence, which we shall denote by  $\mathbf{r} \equiv [r_1, \dots, r_N]$ , may differ from the transmitted one. In order to perform error-correction at the receiver, the key quantity to be considered is the a-posteriori probability, that is, the conditional probability that the transmitted sequence is  $\mathbf{s}$ , given that the received sequence is  $\mathbf{r}$ , which we shall denote as  $p(\mathbf{s}|\mathbf{r})$ . It is a standard exercise of Bayesian inference to show that such a probability can be written as

$$p(\mathbf{s}|\mathbf{r}) \propto \chi(\mathbf{s}) \left( \frac{x}{1-x} \right)^{D(\mathbf{s}, \mathbf{r})}, \quad (1)$$

where  $\chi(\mathbf{s})$  is a characteristic function, taking value 1 if  $\mathbf{s}$  is a codeword and 0 otherwise, and  $D(\mathbf{s}, \mathbf{r})$  is the *Hamming distance* (i.e. the number of different bits) between  $\mathbf{s}$  and  $\mathbf{r}$ . The proportionality symbol stands for a suitable prefactor (depending on  $\mathbf{r}$ ), which ensures the correct normalization of the conditional probability, namely

$$\sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{r}) = 1, \quad (2)$$

the sum running over all possible bit sequences of length  $N$ .

For algebraic convenience, it is useful to represent each bit by an Ising-like spin variable  $s_i, r_i = \pm 1$  (the bit values 0, 1 are mapped respectively to the spin values +1, -1). As previously mentioned, the a-posteriori probability can thus be rewritten in the form of a Boltzmann distribution for an Ising-like model: the transmitted and received “spins” play the role of configuration variables and external fields, respectively, while the parity checks play the role of (multispin) interactions. Let us first observe that, in terms of spin variables, the Hamming distance can be written as

$$D(\mathbf{s}, \mathbf{r}) = \sum_{i=1}^N \frac{1 - r_i s_i}{2}, \quad (3)$$

because the  $i$ -th term of the summation is equal to 0 if  $r_i = s_i$  and 1 otherwise. Furthermore, it is useful to define a function  $V(\mathbf{s})$ , which counts the number of parity checks violated by a generic sequence  $\mathbf{s}$ . In the spin representation, such a function can be written as

$$V(\mathbf{s}) = \sum_{a \in \mathcal{P}} \frac{1 - \prod_{i \in a} s_i}{2}, \quad (4)$$

because the spin product  $\prod_{i \in a} s_i$  takes values  $+1$  or  $-1$ , if the parity check represented by  $a$  is satisfied or not, respectively. With the above definitions, the a-posteriori probability (1) can be rewritten as

$$p(\mathbf{s}|\mathbf{r}) \propto e^{-[D(\mathbf{s}, \mathbf{r}) + JV(\mathbf{s})] \ln \frac{1-x}{x}}, \quad (5)$$

where we have introduced the fictitious parameter  $J \rightarrow \infty$ , whose role is to set at zero the probability of non-codewords, that is, of every sequence  $\mathbf{s}$  such that  $V(\mathbf{s}) > 0$ . Now, making use of (3) and (4), and defining the “Hamiltonian” function

$$H(\mathbf{s}; \mathbf{r}) \triangleq -J \sum_{a \in \mathcal{P}} \prod_{i \in a} s_i - \sum_{i=1}^N r_i s_i, \quad (6)$$

we can easily write

$$D(\mathbf{s}, \mathbf{r}) + JV(\mathbf{s}) = \frac{H(\mathbf{s}; \mathbf{r})}{2} + \text{const}. \quad (7)$$

Note that the semicolon in the expression  $H(\mathbf{s}; \mathbf{r})$  is meant to distinguish the configuration variables  $\mathbf{s}$  from the parameters (external fields)  $\mathbf{r}$ . Note also that the infinitely large coupling constant  $J$  penalizes (infinitely) the “excited states” of each group of spins involved in a multispin interaction, i.e. it prohibits parity-check violation. Defining also

$$\beta \triangleq \frac{1}{2} \ln \frac{1-x}{x}, \quad (8)$$

after (5) and (7) we can finally write a Boltzmann-like probability

$$p(\mathbf{s}|\mathbf{r}) \propto e^{-\beta H(\mathbf{s}; \mathbf{r})}, \quad (9)$$

where  $\beta$  plays the role of (inverse) temperature. In the literature of error-correcting codes, the temperature defined by equation (8) is sometimes called the Nishimori temperature [28].

### 3. Decoding algorithms

In order to minimize the *frame error rate* of the decoder (i.e. the probability that the output sequence  $\mathbf{s}^* \equiv [s_1^*, \dots, s_N^*]$  is different from the transmitted sequence), one has to choose the sequence  $\mathbf{s}^*$  with the maximum a-posteriori (MAP) probability. Unfortunately, this criterion requires us to explore all the codewords, whose number

grows exponentially with  $N$ , giving rise to a computationally hard task. An alternative criterion is to maximize the posterior probability of each single bit, i.e. the marginal

$$p_i(s_i|\mathbf{r}) = \sum_{\mathbf{s}|s_i} p(\mathbf{s}|\mathbf{r}), \quad (10)$$

the sum running over all bit sequences of length  $N$  with the  $i$ -th bit fixed. The  $i$ -th output bit is thus chosen as

$$s_i^*(\mathbf{r}) = \arg \max_{s_i=\pm 1} p_i(s_i|\mathbf{r}). \quad (11)$$

The latter policy, which is the one adopted in the current paper, is usually called bit-MAP [1] or MPM (maximizer of the posterior marginal) [28]. It is evident that, in principle, the computational complexity is still exponential in  $N$ , but in this case one can resort to approximate mean-field methods (such as the Bethe–Peierls approximation), which provide a direct evaluation of the marginals, being amenable to a much more efficient numerical implementation. The marginals can be written in the form of Boltzmann probabilities for single spins interacting with effective fields, namely,

$$p_i(s_i|\mathbf{r}) \propto e^{\beta h_i(\mathbf{r}) s_i}. \quad (12)$$

The sign of each effective field  $h_i$  determines the spin value  $s_i = \pm 1$  having larger probability, so that the MPM criterion (11) can be easily rephrased in terms of effective fields. If  $h_i = 0$ , both spin values have equal probability  $1/2$ , and the output bit is chosen at random.

### 3.1. Bethe–Peierls approximation and belief propagation

In the Bethe–Peierls approximation scheme, the effective fields (which turn out to depend on  $\beta$  as well) are defined implicitly by a set of simultaneous equations, which also include auxiliary unknowns  $u_{a \rightarrow i}$ , called *messages* in the BP jargon [1], associated to the links of the factor graph:

$$h_i = r_i + \sum_{a \ni i} u_{a \rightarrow i}, \quad (13)$$

$$\tanh(\beta u_{a \rightarrow i}) = \tanh(\beta J) \prod_{j \in a \setminus i} \tanh[\beta(h_j - u_{a \rightarrow j})], \quad (14)$$

where the sum runs over all parity checks involving  $i$ , and the product runs over all bits involved in the parity check  $a$  except  $i$ . Note that by plugging equation (13) into (14), one obtains a set of simultaneous recursive equations for the messages  $u_{a \rightarrow i}$  alone, where the received signals  $r_i$  and the noise  $\beta$  play the role of parameters. The iterative self-consistent solution of these equations is an instance of BP (in fact, it can be regarded as a propagation of probabilistic information over the factor graph). The resulting decoding algorithm is described below, the symbol “ $:=$ ” denoting the usual assignment statement.

- (i) Initialization: for  $i = 1, \dots, N$ , set  $h_i := r_i$  and  $u_{a \rightarrow i} := 0 \ \forall a \ni i$ .

- (ii) Evaluate the tentative output sequence  $\mathbf{s}^* \equiv [s_1^*, \dots, s_N^*]$  according to equations (11) and (12), namely

$$s_i^* := \begin{cases} \operatorname{sgn} h_i & \text{if } h_i \neq 0 \\ \pm 1 \text{ at random} & \text{if } h_i = 0 \end{cases} . \quad (15)$$

If  $\mathbf{s}^*$  is a codeword, terminate, otherwise continue.

- (iii) For every parity check  $a \in \mathcal{P}$ , compute a new estimate of the “outgoing” messages according to equation (14), namely

$$\hat{u}_{a \rightarrow i} := \beta^{-1} \tanh^{-1} \prod_{j \in a \setminus i} \tanh[\beta(h_j - u_{a \rightarrow j})] \quad \forall i \in a, \quad (16)$$

where a “hat” indicates that the updated messages are stored in a different memory location. Recall that we are interested in the limit  $J \rightarrow \infty$ , so that the term  $\tanh(\beta J) \rightarrow 1$  can be dropped.

- (iv) For every bit  $i = 1, \dots, N$ , recompute the effective field according to equation (13), using the updated messages, namely

$$h_i := r_i + \sum_{a \ni i} \hat{u}_{a \rightarrow i}, \quad (17)$$

and then “align” the messages values, i.e. assign  $u_{a \rightarrow i} := \hat{u}_{a \rightarrow i} \forall a \ni i$ .

- (v) Go back to step (ii).

Let us note that, as usual for decoding, the stop test of BP (step (ii)) is not based on the convergence of the message values, but rather it requires that the instantaneous values of the effective fields are mapped, according to equation (15), to a valid output sequence (i.e. a codeword).

Possible variants of the above scheme may be addressed to reduce the numerical complexity or to improve convergence, or both. As far as the former issue is concerned, the greatest difficulties arise from the message update statement (16), which requires repeated evaluations of the hyperbolic tangent function (and its inverse) and a number of multiplications. This equation may be replaced by the following, much simpler one

$$\hat{u}_{a \rightarrow i} := \min_{j \in a \setminus i} |h_j - u_{a \rightarrow j}| \prod_{j \in a \setminus i} \operatorname{sgn}(h_j - u_{a \rightarrow j}), \quad (18)$$

derived by taking the limit  $\beta \rightarrow \infty$  (zero temperature). Note that, according to the Nishimori temperature definition (8), high  $\beta$  means low bit-flip probability  $x$  (i.e. high signal-to-noise ratio), which is a typical regime for applications. The resulting algorithm, usually called *min-sum* [1, 2], turns out to be independent of the parameter ( $\beta$  or  $x$ ) which characterizes the noise level. As far as convergence is concerned, one empirically observes that considerable improvement can be obtained by a sequential update strategy, whose basic idea is to use updated message values as soon as they become available [29]. With respect to the previous scheme, such a strategy consists in replacing steps (iii) and (iv) with a unique subroutine, which, for every parity check  $a \in \mathcal{P}$ , performs the

message update (16) and then, for each bit  $i \in a$ , recomputes  $h_i$  according to (17)<sup>†</sup> and immediately executes the alignment statement  $u_{a \rightarrow i} := \hat{u}_{a \rightarrow i}$ . The best results are obtained if the order in which the parity checks are processed is changed for each iteration and taken at random [30]. In the following, we shall denote the latter strategy as *random sequential* (RS) update and, accordingly, the BP algorithm equipped with this strategy will be tagged as RSBP.

### 3.2. Probability damping

Let us now observe that the effective-field update statement (17) can be replaced (without introducing any actual change) by

$$h_i := h_i + \Delta_i, \quad (19)$$

where the difference term  $\Delta_i$  can be computed using either of the following two formulae:

$$\Delta_i := r_i + \sum_{a \ni i} \hat{u}_{a \rightarrow i} - h_i, \quad (20)$$

$$\Delta_i := \sum_{a \ni i} (\hat{u}_{a \rightarrow i} - u_{a \rightarrow i}). \quad (21)$$

Our proposal is to replace equation (19) with the following one

$$h_i := h_i + (1 - \gamma)\Delta_i, \quad (22)$$

where the differential term is attenuated by the factor  $1 - \gamma$ , and  $\gamma \in [0, 1)$  is the adjustable *damping parameter*. For  $\gamma = 0$  we get back an ordinary BP, while increasing  $\gamma$  values decreases the amplitude of the difference term, and progressively slows down the algorithm dynamics. Note that for  $\gamma > 0$  the use of equations (20) or (21) is no longer equivalent. In particular, using (20), the update statement reads

$$h_i := (1 - \gamma) \left( r_i + \sum_{a \ni i} \hat{u}_{a \rightarrow i} \right) + \gamma h_i, \quad (23)$$

namely, each updated field turns out to be a convex linear combination of the “old” value with the “new” value computed by the ordinary BP rule (17). As a consequence, the resulting algorithm turns out to be exactly equivalent to the PDBP algorithm, proposed in [17]. The use of (21) gives rise to a different algorithm (from now on tagged as PD’BP), which, in spite of the apparent similarity with PDBP, turns out to exhibit a considerably different behaviour. To conclude this section, let us note that both PDBP and PD’BP can be implemented either with the “exact” message update rule (16) or with the simplified (min–sum) rule (18). For simplicity, in the following analysis we will consider only the latter case.

<sup>†</sup> In this case the initialization statement  $\hat{u}_{a \rightarrow i} := 0$  must be added at step (i).



#### 4. Numerical results

In this section we shall investigate the performance of the proposed decoding algorithms (PDBP and PD'BP), compared to algorithms using ordinary update equations (BP and RSBP) and compared to the recent DMBP algorithm [26]. Let us stress the fact that all these algorithms, except DMBP, preserve BP fixed points (i.e. stationary points of the Bethe free energy), at odds with other well-known message-passing schemes (for example, so-called *tree-reweighted belief propagation* [31] and *fractional belief propagation* [32]), designed to compute stationary points of modified free energy functionals. Conversely, DMBP perturbs BP fixed points without reference to any variational functional, but rather through a direct modification of the update equations, inspired by the so-called *divide-and-concur* strategy [33]. As far as message scheduling is concerned, let us recall that all of the algorithms analyzed below (except RSBP) implement a parallel update scheme, since our main focus is on achieving practical decoding algorithms, which might be more easily implemented in hardware.

We shall consider a specific code (with block length  $N = 1057$  and rate  $\rho \approx 0.77$ ) available from MacKay's repository [34], and already used in [26] as a test bed for DMBP. This code is (almost) a *regular code*, since every bit is involved in the same number of parity checks  $n = 3$ , while every parity check involves the same number of bits  $m = 13$  (except one check involving 12 bits).

The different algorithms will be characterized primarily in terms of their *frame error rate*. In this respect, let us recall that a decoding failure may occur either when the algorithm does not find a codeword within the prescribed maximum number of iterations, or when it finds a codeword different from the transmitted one. These two kinds of event are denoted respectively as *detected* or *undetected* errors.<sup>‡</sup> The BSC behaviour for each sequence transmission is described by an *error pattern*  $[r_1 s_1, \dots, r_N s_N]$ , which is (in the spin representation) an array containing  $+1$  and  $-1$  values, respectively denoting bits that have been received correctly or incorrectly. In the following, the number of  $-1$  values in the error pattern, i.e. the Hamming distance  $D(\mathbf{s}, \mathbf{r})$ , will be shortly denoted as the *weight* of the error pattern itself. Note that the transmitted sequence  $\mathbf{s}$  is irrelevant to the decoder behaviour, so that simulations can be performed with a unique  $\mathbf{s}$  (for instance  $\mathbf{s} = [+1, \dots, +1]$ ). For each simulation, the number of sampled error patterns is adjusted to obtain a significant number (fixed at 300) of decoding failures.

<sup>‡</sup> In principle, detected errors may be further divided into two categories, namely, actual non-convergence or convergence to a non-codeword. We do not distinguish these two events because, as previously mentioned, the stop test of our algorithm simply checks whether the current values of the effective fields correspond to a codeword, so that convergence to a non-codeword is equivalent to non-convergence.

#### 4.1. Error floor

The frame error rate on a BSC with bit-flip probability  $x$  can be written as

$$P(x) = \sum_{d=d_o}^N \mathcal{N}_d x^d (1-x)^{N-d}, \quad (24)$$

where  $\mathcal{N}_d$  is the number of error patterns of weight  $d$  that cause a decoding failure, and  $d_o$  is the minimum value of  $d$  such that  $\mathcal{N}_d \neq 0$ . The coefficients  $\mathcal{N}_d$  (and therefore also  $d_o$ ) depend on all the parameters that characterize the decoding algorithm, namely, the maximum number of allowed iterations  $\nu$ , the damping parameter  $\gamma$  (if any), and (in principle) the bit-flip probability  $x$ . As mentioned in the previous section, the last parameter is actually irrelevant for min-sum-like algorithms, which implies that the frame error rate  $P(x)$  is simply a polynomial of degree  $N$  (with no terms of degree less than  $d_o$ ). We can thus write

$$P(x) = \sum_{k=d_o}^N \tilde{\mathcal{N}}_k x^k, \quad (25)$$

where the relationships between the two sets of coefficients  $\tilde{\mathcal{N}}_k$  and  $\mathcal{N}_d$  are as follows:

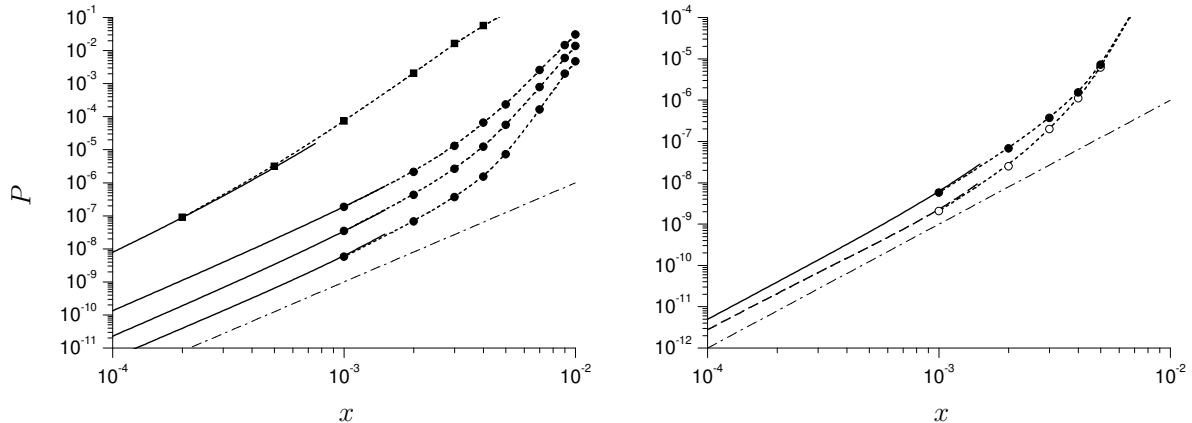
$$\frac{\tilde{\mathcal{N}}_k}{\binom{N}{k}} = \sum_{d=d_o}^k \binom{k}{d} (-1)^{k-d} \frac{\mathcal{N}_d}{\binom{N}{d}} \quad k = d_o, \dots, N, \quad (26)$$

$$\frac{\mathcal{N}_d}{\binom{N}{d}} = \sum_{k=d_o}^d \binom{d}{k} \frac{\tilde{\mathcal{N}}_k}{\binom{N}{k}} \quad d = d_o, \dots, N. \quad (27)$$

In particular, we have  $\tilde{\mathcal{N}}_{d_o} = \mathcal{N}_{d_o}$ , which makes it evident that the asymptotic behaviour of the frame error rate for low  $x$  (i.e. the error floor) is governed by the error patterns of minimum weight (that cause decoding failure). In formulae:

$$P(x) = \mathcal{N}_{d_o} x^{d_o} + o(x^{d_o}). \quad (28)$$

In general, to compute the coefficients  $\tilde{\mathcal{N}}_{d_o}, \tilde{\mathcal{N}}_{d_o+1}, \dots, \tilde{\mathcal{N}}_\ell$  (that is, the  $\ell$ -th order Taylor approximation), one needs to know  $\mathcal{N}_{d_o}, \mathcal{N}_{d_o+1}, \dots, \mathcal{N}_\ell$  (that is, one has to study the decoder behaviour with error patterns of weight up to  $\ell$ ). In the following, we shall describe several cases in which  $d_o = 3$ , and we shall determine the 5-th order Taylor approximation. Thanks to the relatively small size of the code, it is possible to analyze exhaustively the decoder behaviour under all possible error patterns of weight up to  $d = 3$ , and therefore to determine  $\mathcal{N}_3$  exactly. Conversely,  $\mathcal{N}_4$  and  $\mathcal{N}_5$  can be determined approximately by simulations, sampling error patterns of given weight  $d = 4, 5$ . Note that such simulations evaluate the fraction (rather than the total number) of error patterns that cause decoding failure, i.e. the quantity  $\mathcal{N}_d / \binom{N}{d}$ , which directly appears in the conversion formulae (26) and (27).

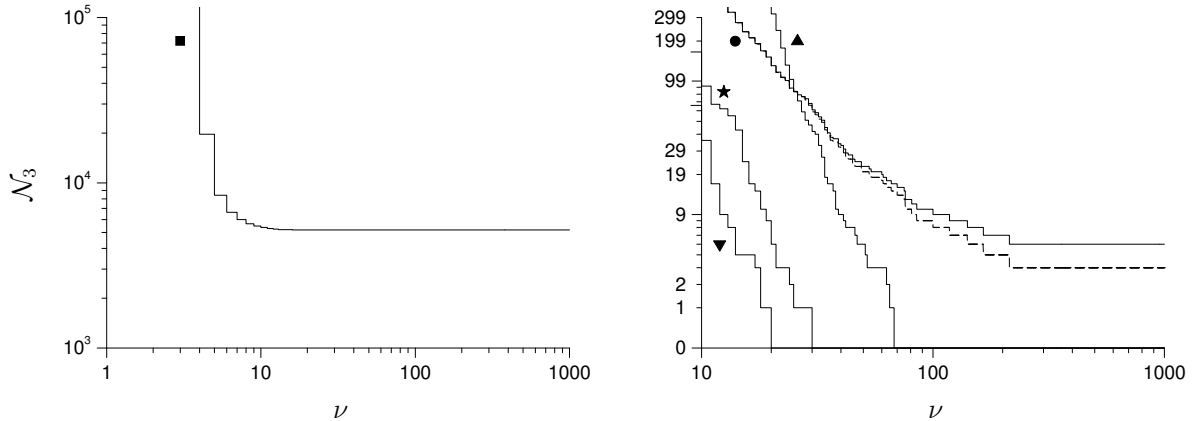


**Figure 1.** Frame error rate  $P(x)$  for BP (squares) and RSBP (circles), with  $\nu = 20, 50, 300$  (left panel) or  $\nu = 300$  (right panel). Solid and hollow symbols denote respectively the total error rate and the one restricted to detected errors. Solid and dashed lines represent the 5-th order Taylor approximations, respectively. Dotted lines are an eye-guide. Thin dash-dotted lines represent  $x^3$ .

#### 4.2. Ordinary BP algorithms

In order to make a better assessment of the improvements that can be achieved using the modified decoding algorithms (PDBP and PD'BP), we first investigated the behaviour of the ordinary min-sum algorithm (BP) and of its variant with a random sequential update (RSBP). The results in terms of frame error rate  $P(x)$  are reported in figure 1, for different values of the maximum number of allowed iterations  $\nu$ . It is noticeable that the performance of BP is practically insensitive to  $\nu$ , in the range of values investigated (the different plots are indistinguishable at the scale of the figure). RSBP yields remarkably better performance, which, in addition, can be improved by increasing  $\nu$ . This effect suggests that the RS update strategy is able to eliminate a large number of trapping sets. Nevertheless, in general, the effect is not quantitatively as relevant as that observed for the code under consideration, and in any case it is not of great importance from a technical point of view, because the RS scheduling cannot be easily parallelized.

We can observe that  $P(x)$  remains asymptotically proportional to  $x^3$  for  $x \rightarrow 0$ , which means, according to the above discussion, that we always have  $d_o = 3$  (so that we have been able to perform the exhaustive analysis). Very low values of  $\nu$  suffice to obtain  $\mathcal{N}_1 = 0$  and  $\mathcal{N}_2 = 0$ , while the behaviour of  $\mathcal{N}_3$  as a function of  $\nu$  is reported in figure 2. It turns out that BP reaches a plateau value  $\mathcal{N}_3 = 5180$  at  $\nu = 21$ , and there is no effect of increasing the maximum number of allowed iterations, at least up to  $\nu = 10^6$ . For RSBP the plateau value is  $\mathcal{N}_3 = 5$ , reached at  $\nu = 214$ , which confirms that the number of iterations is much more effective in this case. Nevertheless, not even RSBP is able to yield  $\mathcal{N}_3 = 0$  (i.e. to provide correct decoding for every error pattern of weight 3), due in particular to the onset of a few undetected errors (the dashed line in figure 2 denotes *detected* errors), which do not occur for parallel BP. This effect is probably related to



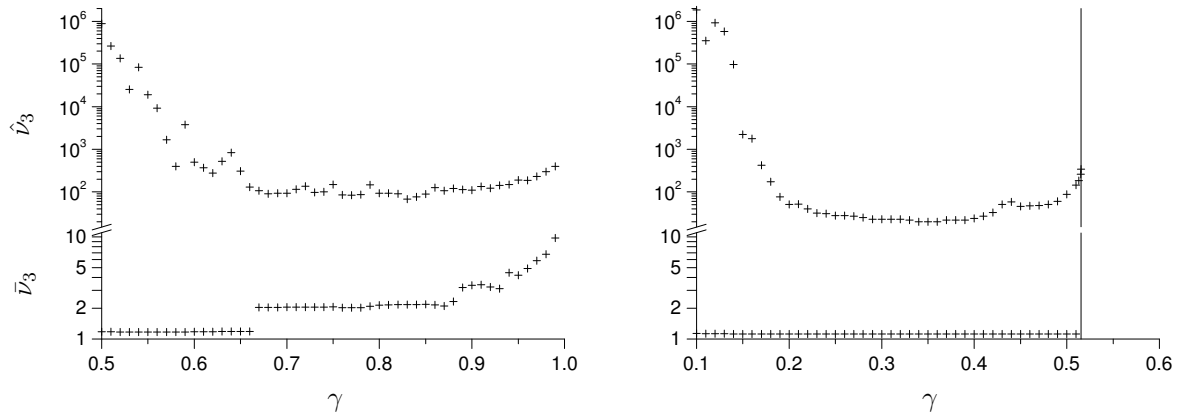
**Figure 2.** Number  $\mathcal{N}_3$  of error patterns of weight 3 that cause decoding failure, generic (solid lines) or detectable (dashed lines), as a function of the maximum number of allowed iterations  $\nu$ , for the following algorithms (each one tagged by a symbol): BP (square), RSBP (circle), PDBP with  $\gamma = 0.83$  (up-triangle), PD'BP with  $\gamma = 0.35$  (down-triangle), DMBP with  $Z = 0.35$  (star) [26]. Note that, in principle, the function  $\mathcal{N}_d(\nu)$  is defined only for integer values of its argument, but for graphical reasons we report the graph of  $\mathcal{N}_d(\lfloor \cdot \rfloor)$  (which is a right-continuous function), “welding” the discontinuities.

the randomized nature of RSBP, which increases its ability to explore the configuration space, and this, on the other hand, is likely to be the same effect that enables it to escape most trapping sets. The relevance of undetected errors to the frame error rate (and in particular to the error floor) of RSBP can be appreciated from the right panel of figure 1. Incidentally, let us observe that for the code under investigation, the minimum Hamming distance between codewords is 8, so that it is not possible, even with an ideal MAP decoder, to achieve an asymptotic slope larger than 4 (i.e.  $\mathcal{N}_4 = 0$ ), because a received sequence with 4 errors can be at equal distance from more than one codeword. The right panel of figure 2 shows that the modified algorithms (namely, PDBP, PD'BP and DMBP, with suitable parameter choices) can achieve the best possible result (i.e.  $\mathcal{N}_3 = 0$ , according to the above argument) for a relatively low number of iterations, as will be detailed in the next section.

#### 4.3. Probability-damping algorithms

Let us now consider PDBP and PD'BP in more detail. We have performed the exhaustive analysis for all error patterns of weight 3 as a function of the damping parameter  $\gamma$  (as well as ordinary BP, these algorithms yield  $\mathcal{N}_1 = \mathcal{N}_2 = 0$  for very low values of  $\nu$ ). The results are reported in figure 3 in terms of the average and maximum number of iterations needed to obtain convergence.<sup>§</sup> These quantities, which we respectively denote by  $\bar{\nu}_3$  and  $\hat{\nu}_3$ , are related to the function  $\mathcal{N}_3(\nu)$  (see figure 2) in

<sup>§</sup> In this case we never encounter undetected errors, so that *convergence* implies *correct decoding*.



**Figure 3.** Average number  $\bar{\nu}_3$  and maximum number  $\hat{\nu}_3$  of iterations needed to obtain convergence for all error patterns of weight 3, for PDBP (left panel) and PD'BP (right panel), as a function of the damping parameter  $\gamma$ . We have reported data points (cross symbols) rather than a continuous interpolating line, because the resolution in  $\gamma$  that we have been able to obtain (with a reasonable computational effort) is not sufficient to faithfully represent the fluctuations of the dependent variables. The solid vertical line marks the threshold  $\gamma$  value for PD'BP (see the text).

a simple way, namely, considering error patterns of a generic weight  $d$ , we have

$$\bar{\nu}_d = \frac{1}{\binom{N}{d}} \sum_{\nu=0}^{\hat{\nu}_d-1} \mathcal{N}_d(\nu), \quad (29)$$

$$\hat{\nu}_d = \min \{ \nu \in \mathbb{N} \mid \mathcal{N}_d(\nu) = 0 \}. \quad (30)$$

Such equations hold if there exists some finite  $\nu$  yielding  $\mathcal{N}_d(\nu) = 0$ , otherwise both  $\bar{\nu}_d$  and  $\hat{\nu}_d$  go to infinity.

We can observe that, for low  $\gamma$  values,  $\hat{\nu}_3$  tends indeed to infinity, which is consistent with the fact that ordinary BP is not able to obtain convergence for all error patterns of weight 3. For increasing  $\gamma$  values,  $\hat{\nu}_3$  first exhibits a decreasing trend, characterized by very large fluctuations, and then a plateau, with relatively smaller fluctuations. Up to this point, both PDBP and PD'BP show the same qualitative behaviour. Some quantitative differences appear both in the  $\gamma$  value at which the plateau begins (namely, about 0.65 for PDBP and about 0.2 for PD'BP) and in the plateau values of  $\hat{\nu}_3$  (roughly 70–150 for PDBP and 20–50 for PD'BP). For larger  $\gamma$  values, the behaviour of the two algorithms becomes even qualitatively different. For PDBP,  $\hat{\nu}_3$  exhibits just a slight increasing trend, but it basically remains finite up to  $\gamma \lesssim 1$ . Conversely, for PD'BP, there is a certain threshold value of  $\gamma$  (slightly above 0.5), such that  $\hat{\nu}_3$  first undergoes a very sharp increase, and then becomes practically infinite (in fact we verify that it is larger than  $10^7$ ). These empirical observations suggest that PDBP and PD'BP, in spite of their similarity, have in fact two very different dynamical behaviours, namely, even though both algorithms are able, at the first stage, to eliminate all possible trapping sets associated to low-weight error patterns, PD'BP appears to introduce new trapping

sets upon increasing  $\gamma$ . On the contrary, the worsening performance of PDBP for large growing  $\gamma$  values is likely to be ascribed only to the fact that the algorithm dynamics get slower and slower in this regime. This conclusion is supported by the fact that, in this regime, even the average number of iterations  $\bar{\nu}_3$  gets larger and larger. It is also noticeable that, in the low  $\gamma$  regime, in which the *maximum* number of iterations  $\hat{\nu}_3$  is still very large, the *average* number of iterations  $\bar{\nu}_3$  is just slightly larger than 1, which means that for most error patterns of weight 3, PDBP converges in 1 iteration (or a little more), while there is a very small fraction of instances, which require a huge number of iterations to converge. Conversely, at the beginning of the plateau region of  $\hat{\nu}_3$  (precisely at  $\gamma = 2/3$ ),  $\bar{\nu}_3$  exhibits an abrupt jump to slightly above 2 (meaning that most instances converge in 2 iterations), and in particular one can observe that, for  $\gamma \geq 2/3$ , absolutely no instances converge in 1 iteration. The latter fact can be easily rationalized, taking into account that, at the first iteration, we have  $h_i = r_i$  and  $u_{a \rightarrow i} = 0$ . As a consequence, the effective field update statement (23) becomes

$$h_i := r_i + (1 - \gamma) \sum_{a \ni i} \hat{u}_{a \rightarrow i}, \quad (31)$$

with  $|\hat{u}_{a \rightarrow i}| = 1$ , due to the message update statement (18). Now, since the number of summed messages in the above equation is  $n$  (with  $n = 3$  for the current code), if

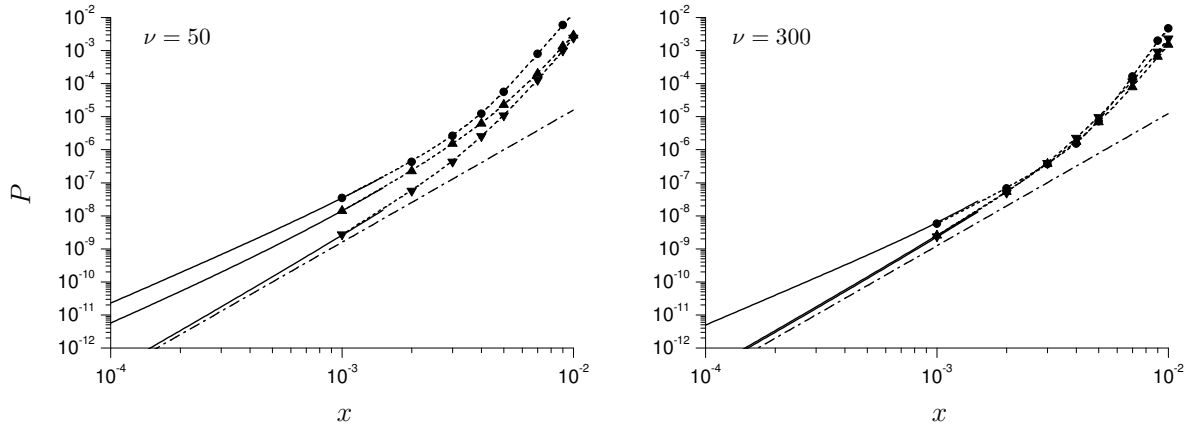
$$\gamma \geq 1 - \frac{1}{n}, \quad (32)$$

there is no chance to correct an incorrect sign of  $r_i$  at the first iteration. Further similar “phase transitions” can be detected in the large  $\gamma$  regime, where one can progressively observe no instances converging within 2, 3, 4, ... iterations. As far as PD’BP is concerned, the behaviour of  $\bar{\nu}_3$  is completely different, namely,  $\bar{\nu}_3$  stays very close to 1 in the whole region where  $\hat{\nu}_3$  remains finite. This is of course a good fact for the decoder performance. Nevertheless, the presence of a region in which  $\hat{\nu}_3$  becomes infinite at high  $\gamma$ , as described above, suggests that the damping mechanism introduced by PD’BP might be less robust, with respect to PDBP, and that in any case, it requires a finer tuning of the damping parameter.

By the way, let us recall that in the framework of the interpretation of PDBP as an approximation of a double-loop algorithm [16], the damping parameter can be related to a set of *allocation coefficients*, which are associated to the edges of the factor graph and allow one to define a sufficient condition for convergence [17]. Assuming that for a regular factor graph (i.e. a regular code) all the allocation coefficients are equal, one can map the aforementioned condition to the following simple inequality:

$$\gamma \geq 1 - \frac{1}{n} \left( 1 - \frac{1}{m} \right)^{-1}. \quad (33)$$

The latter can no longer be proved to be a sufficient condition for convergence of PDBP, but we believe that it can be used as a “rule of thumb” to identify a range of values of the damping parameter for which the algorithm works best. Note that, for  $n = 3$  and  $m = 13$ , condition (33) reads  $\gamma \geq 0.63\bar{8}$ , which, quite surprisingly, roughly corresponds



**Figure 4.** Frame error rate  $P(x)$  for RSBP (circles), PDBP with  $\gamma = 0.83$  (up-triangles) and PD'BP with  $\gamma = 0.35$  (down-triangles), for  $\nu = 50$  (left panel) and  $\nu = 300$  (right panel). The solid lines represent the 5-th order Taylor approximations. The dotted lines are an eye-guide. The dash-dotted lines represent the asymptotic behaviour (4-th order approximation) for DMBP with  $Z = 0.35$ .

to the plateau region of  $\hat{\nu}_3$ , described above (see the left panel of figure 3). A similar rule is not available for PD'BP, since we do not have an analogous heuristic interpretation for the latter algorithm.

As far as DMBP is concerned, one can observe some similarities to PD'BP, with an even higher sensitivity to the tuning parameter  $Z$  (see [26] for the precise definition). In particular,  $\hat{\nu}_3$  remains finite (with values roughly in the range 30–70) only for  $0.293 \lesssim Z \lesssim 0.433$ , the latter interval being delimited on both sides by “sharp thresholds”, like the one displayed by PD'BP. In this interval, the average number of iterations  $\bar{\nu}_3$  is almost constant and slightly less than 2. One can also verify that the high- $Z$  threshold is characterized by the onset of a real non-convergence (in principle we can only state that  $\hat{\nu}_3$  becomes larger than  $10^7$ ), while the low- $Z$  threshold is characterized by convergence to a non-codeword. The latter behaviour is likely to be ascribed to the considerable distortion of DMBP dynamics with respect to pure BP. In particular, at odds with PDBP and PD'BP, DMBP does not preserve BP fixed points (except for  $Z = 0.5$ ), as previously mentioned. Just to give an order of magnitude for the phenomenon of convergence to non-codewords, we have analyzed all the failure events collected in our simulations with error-pattern weight up to 5. We observed that, for DMBP, 100% of detected errors are due to non-codewords, while, conversely, no such event occurs for pure BP. The percentage is rather large even for RSBP (about 75%) and PD'BP (about 55%), while, remarkably, PDBP retains the pure BP behaviour, with precisely 0% non-codewords.

In the remainder of this section, we discuss the performance of PDBP and PD'BP in terms of the frame error rate. The damping parameter values (respectively  $\gamma = 0.83$  and  $\gamma = 0.35$ ) are chosen to minimize  $\hat{\nu}_3$ , according to the previous analysis (from which we

obtain respectively  $\hat{\nu}_3 = 68$  and  $\hat{\nu}_3 = 20$ , see figure 3). The results, reported in figure 4, are compared to those of RSBP and of DMBP with the best-performing value of the tuning parameter  $Z = 0.35^*$  [26]. The behaviour of all these cases in terms of  $\mathcal{N}_3(\nu)$  has been reported in figure 2, showing that PD'BP takes the lowest number of iterations to correct all error patterns of weight 3. For  $\nu = 50$  (figure 4, left panel), we observe that PDBP slightly improves the error floor of RSBP, but the asymptotic slope is still 3, because, according to figure 2, we still have  $\mathcal{N}_3 > 0$  (i.e. the algorithm is not yet able to correct all weight-3 error patterns). Conversely, for PD'BP the asymptotic slope is 4, because  $\mathcal{N}_3 = 0$  (still according to figure 2). In these conditions, the error floor behaviour of PDBP is outperformed by PD'BP, whose frame error rate turns out to be almost coincident with that of DMBP. To avoid confusion, for the latter algorithm we have reported only the asymptotic behaviour (4-th order approximation), because the full graph turns out to be almost completely overlapped with that of PD'BP. Upon increasing the maximum number of allowed iterations (in the right panel of figure 4 we report the case  $\nu = 300$ ), the error rates of PDBP and PD'BP become more and more similar to each other, with asymptotic slope 4, and both are practically equivalent to that of DMBP.

## 5. Conclusions and perspectives

In this paper we have considered a BP algorithm with over-relaxed dynamics, denoted as probability-damping belief propagation (PDBP) [17], and a variant of latter (PD'BP), employed as decoding algorithms for low-density parity-check (LDPC) codes. In terms of the frame error rate, it turns out that PD'BP matches the performance of difference-map belief propagation (DMBP) [26], which is one of the best-performing iterative decoding algorithms of the BP type (and of comparable complexity), proposed in the last few years. Furthermore PDBP achieves basically an equivalent performance, but only with a larger number (e.g.,  $\sim 100$ ) of allowed iterations. The particular code studied, previously used as a test bed for DMBP, has a minimum distance of 8, so that any decoder starts making errors when 4 bits have been flipped. Therefore, we have carefully investigated whether decoders can successfully correct all error patterns of weight 3. It turns out that BP and RSBP do not successfully correct all such patterns, while PDBP, PD'BP, and DMBP do. In particular, PD'BP required the fewest number of iterations to achieve guaranteed success, followed by DMBP, and then followed by PDBP. Since practical decoders must limit the number of iterations to get an acceptable throughput, this is a significant result, meaning that PD'BP is (at least in this case) the most practical decoder. We recognize that a weak point in our investigation is the fact that it is based on the analysis of a single specific code, characterized by a specially relevant error floor, and on an exceedingly simple model of a noisy channel (BSC), so it is not clear whether our claims may hold in a more general setting. Nevertheless, preliminary

\* The equality of this value with the optimal  $\gamma$  value for PD'BP is a mere coincidence, because the two parameters have in fact a different meaning.



(unpublished) results of further simulations suggest that most of the observations and arguments developed in this article can be extended to a wider variety of LDPC codes, and that considerable performance improvements can be obtained even with a more realistic (e.g. Gaussian) channel model.

One original feature of PDBP and PD'BP, compared to different BP-inspired decoding algorithms [35, 36], including DMBP, is that they preserve BP fixed points. As an important consequence, we expect they can be used for a much broader spectrum of problems, in particular when it is important to actually compute the Boltzmann marginal probabilities (in the Bethe–Peierls approximation), and convergence difficulties with pure BP are encountered [18, 19]. An additional feature of PDBP, which we have emphasized in the text, is that one can regard it as an approximation of a provably-convergent double-loop algorithm [16]. As a consequence, we expect that its more favourable convergence properties might be quite robust, compared to other algorithms such as PD'BP itself, whose over-relaxing rule has only been devised by analogy. In particular, we expect that PDBP might perform even better than PD'BP on codes that have already been optimized to reduce the incidence of trapping sets [23, 24].

Finally, we would like to stress two more facts concerning PDBP, which in our opinion make it rather appealing from a practical point of view. First, it provides a heuristic rule to choose the value of the damping parameter. Secondly, it shows very low sensitivity to variations of this value, at odds with PD'BP, and especially DMBP. The former fact might be very important for the design of a real decoder. Furthermore, looking at the original paper in which PDBP was proposed [17], one can argue that, in the case of irregular factor graphs, the aforementioned rule can be generalized with non-uniform (i.e. bit-dependent) damping parameters. We expect that such a generalization might provide further improvements of the error floor for irregular codes [36]. As far as the second fact is concerned, this may even be very important in a practical implementation, because it should guarantee that the algorithm performance is not affected by quantization. Of course, a more detailed assessment of both of these issues is far beyond the scope of the present paper, as it would require considerable extra work, but it will probably be the subject of future research.

## Acknowledgments

I gratefully acknowledge the useful suggestions and discussions from and with R Zecchina, A Braunstein, F Kayhan and G Montorsi.

## References

- [1] Mézard M and Montanari A 2009 *Information, Physics and Computation* (Oxford University Press)
- [2] Yedidia J S 2011 *J. Stat. Phys.* **145** 860
- [3] Yedidia J S 2001 *Advanced Mean Field Methods: Theory and Practice* ed M Oppor and D Saad (Cambridge, MA: MIT Press) p 21
- Yedidia J S 2000 *Mitsubishi Electric Technical Report* 2000-27

- [4] Pelizzola A 2005 *J. Phys. A: Math. Gen.* **38** R309
- [5] Richardson T and Urbanke R 2003 *IEEE Commun. Mag.* **41** 126
- [6] Gallager R 1962 *IRE Trans. Inform. Theory* **7** 21
- [7] Kabashima Y and Saad D 2004 *J. Phys. A: Math. Gen.* **37** R1
- [8] Bethe H A 1935 *Proc. R. Soc. A* **150** 552
- [9] Peierls R 1936 *Proc. R. Soc. A* **154** 207
- [10] Burley D M 1972 *Phase Transitions and Critical Phenomena* vol 2 ed C Domb and M S Green (New York: Academic) chapter 9
- [11] Kabashima Y and Saad D 1998 *Europhys. Lett.* **44** 668
- [12] Yedidia J S, Freeman W T and Weiss Y 2001 *Mitsubishi Electric Technical Report* 2001-16
- [13] Frey B J and MacKay D J C 1998 *Proc. 10th Conf. on Advances in Neural Information Processing Systems (NIPS-97, Denver, CO, 1-6 December 1997)* (Cambridge, MA: MIT Press) p 479
- [14] An G 1988 *J. Stat. Phys.* **52** 727
- [15] Heskes T 2003 *Proc. 15th Conf. on Advances in Neural Information Processing Systems (NIPS-02, Vancouver, Canada, 9-12 December 2002)* (Cambridge, MA: MIT Press) p 359
- [16] Heskes T, Albers K and Kappen B 2003 *Proc. 19th Annual Conf. on Uncertainty in Artificial Intelligence (UAI-03, Acapulco, Mexico, 7-10 August 2003)* (San Fransisco, CA: Morgan Kaufmann) p 313
- [17] Pretti M 2005 *J. Stat. Mech.* P11008
- [18] Som P, Datta T, Srinidhi N, Chockalingam A and Rajan B S 2011 *IEEE J. Sel. Top. Signal Process.* **5** 1497
- [19] Wang S, Li Y, Gao Z and Wang J 2013 *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-13, Vancouver, Canada, 26-31 May 2013)* p 5865
- [20] MacKay D J C and Postol M S 2003 *Electron. Notes Theor. Comput. Sci.* **74** 97
- [21] Richardson T 2003 *Proc. 41st Annual Allerton Conf. on Communication, Control and Computing (Monticello, IL, 1-3 October 2003)* p 1426
- [22] Stepanov M G, Chernyak V, Chertkov M and Vasic B 2005 *Phys. Rev. Lett.* **95** 228701
- [23] Hu X-Y, Eleftheriou E and Arnold D M 2005 *IEEE Trans. Inform. Theory* **51** 386
- [24] Asvadi R, Banihashemi A H and Ahmadian-Attari M 2011 *IEEE Trans. Inform. Theory* **57** 2213
- [25] Han Y and Ryan W E 2009 *IEEE Trans. Commun.* **57** 1663
- [26] Yedidia J S, Wang Y and Draper S C 2011 *IEEE Trans. Inform. Theory* **57** 786
- [27] Kschischang F R, Frey B J and Loeliger H-A 2001 *IEEE Trans. Inform. Theory* **47** 498
- [28] Nishimori H 2001 *Statistical Physics of Spin Glasses and Information Processing: an Introduction* (Oxford University Press)
- [29] Sharon E, Litsyn S and Goldberger J 2004 *Proc. 23rd IEEE Convention of Electrical and Electronics Engineers in Israel (Eilat, Israel, 6-7 September 2004)* p 223
- [30] Braunstein A, Mézard M and Zecchina R 2005 *Random Struct. Algorithms* **27** 201
- [31] Wainwright M J, Jaakkola T S and Willsky A S 2002 *Proc. 18th Annual Conf. on Uncertainty in Artificial Intelligence (UAI-02, Edmonton, Canada, 1-4 August 2002)* (San Fransisco, CA: Morgan Kaufmann) p 536
- [32] Wiegner W and Heskes T 2003 *Proc. 15th Conf. on Advances in Neural Information Processing Systems (NIPS-02, Vancouver, Canada, 9-12 December 2002)* (Cambridge, MA: MIT Press) p 438
- [33] Gravel S and Elser V 2008 *Phys. Rev. E* **78** 036706
- [34] MacKay D J C 1998 *Encyclopedia of Sparse Graph Codes*  
([www.inference.phy.cam.ac.uk/mackay/codes/data.html](http://www.inference.phy.cam.ac.uk/mackay/codes/data.html)) code 1057.244.3.457
- [35] Chen J, Dholakia A, Eleftheriou E, Fossorier M P C, Mihaljević M and Hu X-Y 2005 *IEEE Trans. Commun.* **53** 1288
- [36] Chen J, Tanner R M, Jones C and Li Y 2005 *Proc. Int. Symp. on Information Theory (ISIT-05, Adelaide, Australia, 4-9 September 2005)* p 449